AMENDMENTS TO THE SPECIFICATION

Amend the paragraph starting at page 1, line 24, with the following replacement paragraph.

While the system disclosed by Ban in United States Patent No 5,404,485 has certain advantages over previously existing techniques, there are also various limitations associated with these techniques. For example, the described system is relatively complicated to implement. As a virtual memory map is used, the use of standard file-system utilities to debug/repair the stored file-system is precluded if the file-system data is uploaded from an embedded device that uses flash memory and moved to a customer support site. Also, additional data structures need to be stored in the flash memory.

Amend the paragraph starting at page 6, line 34, with the following replacement paragraph.

~~A block in a sector cache is dirty if it is different from what it was, when it was swapped in from the flash disk. This can be tracked by using one bit per block of data. All tracking bits are initially reset. Whenever a block is written to, the corresponding bit is set. The number of dirty blocks for a sector cache is incremented, if the block for which the write request is received has not been already marked dirty.~~ A flash sector consists of multiple file-system blocks. The blocks are identified by a 'sector offset' shown in Figure 3A. A sector weightage is dynamically assigned to each sector that is cached. This count is incremented whenever there is a write access to a file-system block within this sector cache, if the said block has not already caused the sector weightage to be incremented. This can be tracked by using one bit per block of data. All tracking bits are initially reset. Whenever a block is written to or dirtied, the corresponding bit is set. As will be explained later, these tracking bits, as well as the sector

weightage are also reset when the algorithm determines that this sector has suffered enough sector switches to be offered preferential treatment.

Amend the paragraph starting at page 7, line 25, with the following replacement paragraph.

Whenever the number of sector switches to a sector cache reaches a threshold, the sector weightage ~~count of the number of dirtied blocks~~ for that sector cache is reset to zero. When the threshold is reached, the number of sector switches for the sector cache, is also reset to zero. This means that, this sector cache enjoys an advantage compared to other sector caches for subsequent swap decisions, since a sector cache with a greater number of dirtied blocks is swapped-out first.

Amend the paragraph starting at page 7, line 32, with the following replacement paragraph.

If the sector weightage ~~count of dirtied blocks~~ is reset to zero for a sector cache, the sector cache can remain in the sector cache list, until the sector weightage ~~dirty block count~~ increases, and exceeds the sector weightage ~~dirty block count~~ of all other sector caches. Thus, when a sector cache demonstrates the characteristics of a sector cache which holds meta-data information (that is, by having a large number of switches to the sector cache) the sector cache is rewarded with this advantage.

Amend the paragraph starting at page 11, line 8, with the following replacement paragraph.

When the flash-disk device driver receives a request to write a block of data (a write request), the driver performs a sequence of steps. These steps, and the algorithm governing these steps in described below with reference to Figs. 3A and 3B. Figs. 3A and 3B jointly

represent a flowchart that describes the steps for writing to flash-disk using the sector cache list. The write request is serviced as follows:

1. Convert the generated file-system offset, to the tuple (bank number, sector number, sector offset) (step 305). If the file-system image is not contiguous in memory (refer to the section above entitled "Multiple file-systems"), perform the required extra translation (step 310).

2. Check if this sector is in the sector cache list (step 315).

3. If yes (that is, if the checked sector is in the sector cache list),

   (a) Write to the corresponding sector cache and flag the corresponding block as dirty (step 320).

   (b) Increment the sector weightage ~~number of dirty blocks~~ for the sector cache, if that block was not already dirty (step 325).

   (c) Increment the number of switches to this sector cache, if this write involved a sector switch (step 330).

   (d) If the number of sector switches has exceeded the calculated threshold (refer to the section entitled "Detecting sectors holding meta-data"), reset the sector weightage ~~dirty block~~ count and the number of sector switches for this sector cache (steps 335 and 340). Also flag the file-system blocks in the said sector as non-dirty irrespective of whether the contents of the blocks differ from the corresponding flash memory locations.

4. Else,

(a) If there is a free sector cache in the sector cache list, select that sector cache (step 345).

(b) Else,

(1) Traverse the sector cache list to select a sector cache that can be swapped-out to the flash-disk. For this selection, choose a non-pinned sector cache, that has the highest sector weightage ~~largest number of dirty blocks~~ (step 350).

(2) Erase the selected sector (in the appropriate bank) that corresponds to the sector cache chosen in the previous step (step 355). (The erase/wait/write commands use the appropriate register locations in the corresponding bank).

(3) Swap-out the contents of the selected sector cache to flash-disk (step 360).

(c) Swap-in the data from the flash sector for which the write request was received, to the selected sector cache (step 365).

(d) Update data structure entries so that the swap out/in is appropriately reflected (step 370). (refer to the section entitled "Implementation" in this respect).

Amend the paragraph starting at page 13, line 15, with the following replacement paragraph.

Fig. 4 schematically represents a sector map array for the sector cache, stored and represented as a linked list. Each sector cache 410 has an associated sector cache information (**`sc_info`**) structure, which contains the following information listed below. This field of the sector cache information (**`sc_info`**) structure 420 is represented in Fig. 4, with the field names

represented in Fig. 5. The field names of the sector cache information (**sc_info**) structure are described in the list below.

1. Whether the associated sector cache is pinned (**is_pinned**).
2. Sector Cache Weightage (**sector_weightage**) ~~Number of dirty blocks in the sector cache (num_dirty_blocks)~~.
3. Number of sector switches to this sector cache (**num_sector_switch**).
4. The corresponding bank number of the flash-disk sector that it is caching (**bank_number**).
5. The corresponding sector number of the flash-disk sector that it is caching (**sector_number**).
6. Size of this sector (**sector_size**).
7. Address of the next sector cache in the list (**next_sector_cache**).
8. Address of the sector cache (**sector_cache_address**).

Amend the paragraph starting at page 14, line 3, with the following replacement paragraph.

For a write operation to a flash sector, direct indexing is done onto the sector map array, to determine whether it is currently in the sector cache list. If it is (that is, the corresponding entry in the map array is non-NULL), the corresponding sector cache is used for the flash write. Otherwise the sector cache information **(sc_info)** list is traversed in order to determine whether a free cache entry is available in the list (this is done by checking if a pre-determined invalid value is found in the bank/sector number field of any **sc_info** element in the list). Also during this traversal, the address of the **sc_info** element with the highest sector weightage ~~largest number of dirtied blocks~~ is found and stored, provided that a free **sc_info** element has not previously been found. If the sector is not in the cache list and a free **sc_info**

element is available in the list (as determined in the last step), the free entry is populated with the corresponding flash sector's contents and the flash-write is performed onto that sector. If there is no free cache element, the **sc_info** element with the highest sector weightage ~~largest number of dirty words~~ (as determined in the last step) is selected to be swapped back to the flash-disk and thus the freed entry is populated with the corresponding flash-sector and the write is performed onto this sector. The corresponding **sc_info** element is updated with new values. The corresponding pointers in the map array for the old and new sector, are also updated.

Amend the paragraph starting at page 17, line 24, with the following replacement paragraph.

The described technique relates to an algorithm for emulating a disk, in flash memory. The design aims to minimize erases and writes to the flash. This is a solution at the device driver level and is transparent to the file-system. This technique defines a policy to make use of a list of sector caches to buffer accesses to the flash-disk by recording certain characteristics of data access to the file-sytem blocks constituting the cached sectors. The technique also implements a procedure that attempts to detect the sectors that hold critical meta-data information, and gives preferential treatment to the cached sectors corresponding to those detected sectors.